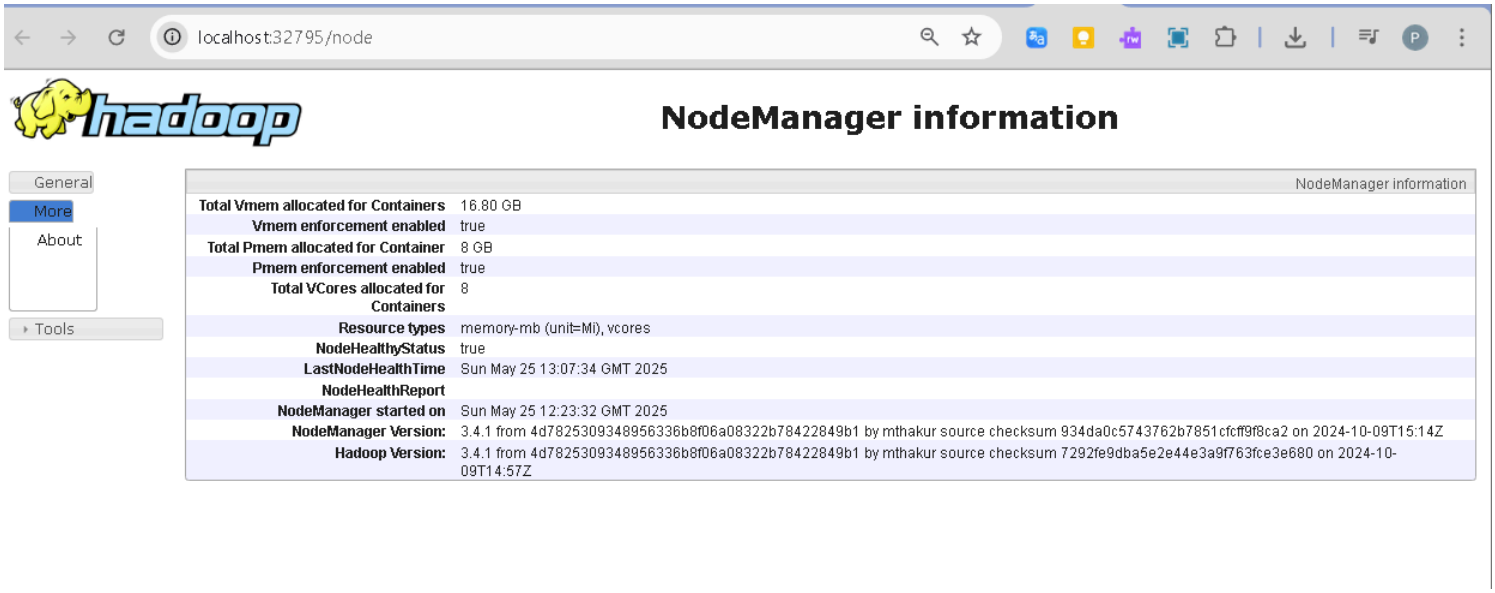
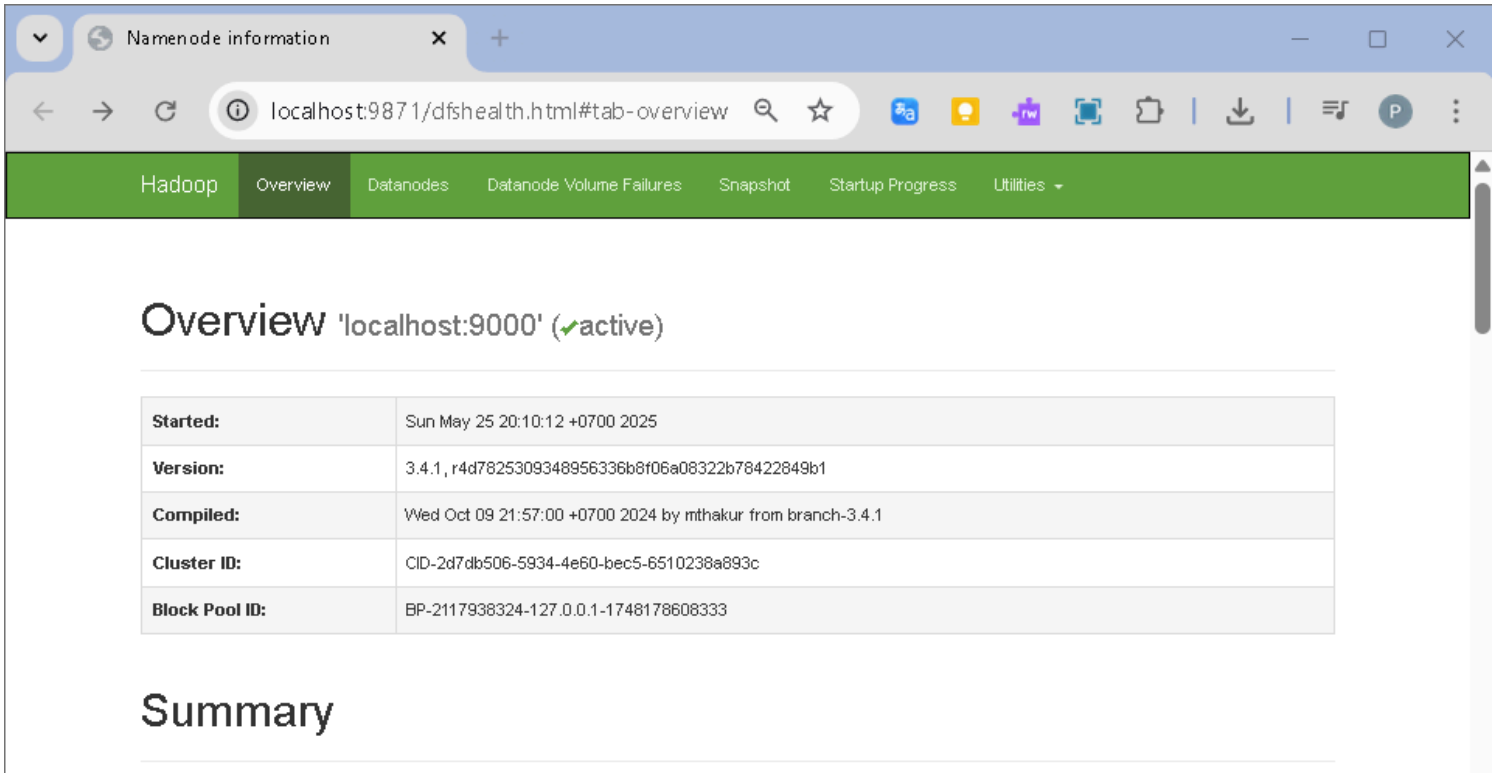
IMPLEMENTASI





STEP 1: Persiapan Awal - Copy File ke Container
1.1 Copy file CSV ke dalam container Hadoop
Buka terminal/command prompt baru
Copy file CSV ke container
docker cp "E:\bigdata-hadoop\data-sumber\data_pertanian_sumatera_3juta.csv" bigdata-hadoop:/tmp/
docker cp "E:\bigdata-hadoop\data-sumber\luas_panen_sumatera_2023_2025.csv"

bigdata-hadoop:/tmp/

```
pardi@PardiSitorus:/mnt/e/bigdata-hadoop$ docker run -v /mnt/e/bigdata-hadoo
p/data-sumber:/data bigdata-hadoop
pardi@PardiSitorus:/mnt/e/bigdata-hadoop$ docker exec -it bigdata-hadoop bas
h
root@localhost:/# ls -la /tmp/*.csv
-rwxrwxrwx 1 ubuntu ubuntu 64699780 May 24 06:57 /tmp/data_pertanian_sumater
a_3juta.csv
-rwxrwxrwx 1 ubuntu ubuntu    21047 May 24 07:46 /tmp/luas_panen_sumatera_20
23_2025.csv
root@localhost:/#
```

1.2 Masuk ke container Hadoop
docker exec -it bigdata-hadoop bash
1.3 Cek file sudah ada
ls -la /tmp/*.csv+

STEP 2: HDFS Setup (Bronze Layer) - Ingesti Data
2.1 Buat direktori di HDFS
Buat direktori bronze (raw data)
hdfs dfs -mkdir -p /bigdata/bronze/pertanian
hdfs dfs -mkdir -p /bigdata/silver/pertanian
hdfs dfs -mkdir -p /bigdata/gold/pertanian

```
root@localhost:/# # Buat direktori bronze (raw data)
hdfs dfs -mkdir -p /bigdata/bronze/pertanian
hdfs dfs -mkdir -p /bigdata/silver/pertanian
hdfs dfs -mkdir -p /bigdata/gold/pertanian
```

Cek direktori berhasil dibuat
hdfs dfs -ls /bigdata/pertanian/

```
# Cek file berhasil diupload
hdfs dfs -ls /bigdata/bronze/pertanian/
Found 2 items
-rw-r--r--   1 root supergroup   64699780 2025-05-25 00:54 /bigdata/bronze/p
ertanian/data_pertanian_sumatera_3juta.csv
-rw-r--r--   1 root supergroup      21047 2025-05-25 00:54 /bigdata/bronze/p
ertanian/luas_panen_sumatera_2023_2025.csv
root@localhost:/#
```

2.2 Upload file CSV ke HDFS (Bronze Layer)
Upload file ke Bronze Layer
hdfs dfs -put /tmp/data_pertanian_sumatera_3juta.csv /bigdata/bronze/pertanian/
hdfs dfs -put /tmp/luas_panen_sumatera_2023_2025.csv /bigdata/bronze/pertanian/

```
root@localhost:/# # Upload file ke Bronze Layer
hdfs dfs -put /tmp/data_pertanian_sumatera_3juta.csv /bigdata/bronze/pertani
an/
hdfs dfs -put /tmp/luas_panen_sumatera_2023_2025.csv /bigdata/bronze/pertani
an/

# Cek file berhasil diupload
hdfs dfs -ls /bigdata/bronze/pertanian/
Found 2 items
-rw-r--r--   1 root supergroup   64699780 2025-05-25 00:54 /bigdata/bronze/p
ertanian/data_pertanian_sumatera_3juta.csv
-rw-r--r--   1 root supergroup      21047 2025-05-25 00:54 /bigdata/bronze/p
ertanian/luas_panen_sumatera_2023_2025.csv
root@localhost:/#
```

Cek file berhasil diupload
hdfs dfs -ls /bigdata/bronze/pertanian/
STEP 3: ETL dengan Spark (Silver Layer) - Transformasi Data
3.1 Masuk ke Spark container
Buka terminal baru
docker exec -it bigdata-spark bash
3.2 Jalankan PySpark
pyspark --master yarn
3.3 Script ETL di PySpark (Copy paste satu per satu)

# Import libraries
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *

# Baca data dari HDFS Bronze Layer
df_pertanian = spark.read.option("header", "true").option("inferSchema",
"true").csv("hdfs://namenode:9000/bigdata/bronze/pertanian/data_pertanian_sumatera_3juta
.csv")

# Lihat struktur data
df_pertanian.printSchema()
df_pertanian.show(5)

# Cleaning data sederhana - hapus null values
df_clean = df_pertanian.dropna()

# Transformasi sederhana - tambah kolom produktivitas
df_transformed = df_clean.withColumn("produktivitas", col("produksi_ton") /
col("luas_panen_ha"))

```
# Filter data yang masuk akal (produktivitas > 0)
df_final = df_transformed.filter(col("produktivitas") > 0)

# Simpan ke Silver Layer
df_final.coalesce(1).write.mode("overwrite").option("header",
"true").csv("hdfs://namenode:9000/bigdata/silver/pertanian/data_clean")

# Cek hasil
print("Data berhasil dibersihkan dan disimpan ke Silver Layer")
df_final.count()
```

STEP 4: Machine Learning dengan Random Forest

```
=== MULAI ETL PROCESS ===
1. Membaca data dari Bronze Layer di: /mnt/e/bigdata-hadoop
/data-sumber/luas_panen_sumatera_2023_2025.csv
Total record awal: 147
Schema data:
root
 |-- Provinsi: string (nullable = true)
 |-- Kabupaten/Kota: string (nullable = true)
 |-- Luas Panen 2018: string (nullable = true)
 |-- Luas Panen 2019: string (nullable = true)
 |-- Luas Panen 2020: string (nullable = true)
 |-- Luas Panen 2021: string (nullable = true)
 |-- Luas Panen 2022: string (nullable = true)
 |-- Luas Panen 2023: string (nullable = true)
 |-- Luas Panen 2024: string (nullable = true)
 |-- Produksi Padi 2018: string (nullable = true)
 |-- Produksi Padi 2019: string (nullable = true)
 |-- Produksi Padi 2020: string (nullable = true)
 |-- Produksi Padi 2021: string (nullable = true)
 |-- Produksi Padi 2022: string (nullable = true)
 |-- Produksi Padi 2023: string (nullable = true)
 |-- Produksi Padi 2024: string (nullable = true)

Contoh data (5 baris pertama):
+--------------+----------------+--------------+----------
-----+--------------+--------------+--------------+-----
----------+--------------+-----------------+------------
-----+-----------------+-----------------+--------------
---+-----------------+-----------------+
|Provinsi      |Kabupaten/Kota  |Luas Panen 2018|Luas Panen
 2019|Luas Panen 2020|Luas Panen 2021|Luas Panen 2022|Luas
```

```
Contoh data (5 baris pertama):
+------------------------+-----+---------+-----------+-------------+------
--------+-----------+------------+
|provinsi                |tahun|komoditas|produksi_ton|luas_panen_ha|curah_
hujan_mm|kelembapan_%|suhu_celsius|
+------------------------+-----+---------+-----------+-------------+------
--------+-----------+------------+
|Kepulauan Riau          |2022 |beras    |974284.25   |99156.02     |2236.5
6       |97.19       |26.1        |
|Aceh                    |2024 |padi     |498215.64   |73460.34     |2547.3
4       |73.18       |25.97       |
|Lampung                 |2023 |beras    |696154.07   |89969.71     |3026.4
3       |78.77       |33.47       |
|Kepulauan Bangka Belitung|2014 |beras    |378497.49   |20496.12     |2227.2
2       |43.15       |23.22       |
|Lampung                 |2023 |beras    |820231.14   |19191.46     |4998.4
7       |51.3        |22.32       |
+------------------------+-----+---------+-----------+-------------+------
--------+-----------+------------+
only showing top 5 rows
2. Membersihkan data (drop baris dengan nilai NULL)...
[Stage 6:===>                                                    (1 + 15) /
[Stage 6:==========>                                             (3 + 13) /

Record setelah cleaning: 1048575 (hilang 0 record)
3. Menambah kolom 'produktivitas' = produksi_ton / luas_panen_ha...
Filter data dengan produktivitas > 0...
[Stage 9:===>                                                    (1 + 15) /

Record final setelah filter produktivitas > 0: 1048575 (hilang 0 record)
4. Menyimpan hasil ke Silver Layer di: /mnt/e/bigdata-hadoop/silver/pertania
n/data_clean
[Stage 12:>                                                       (0 + 1)

=== ETL PROCESS SELESAI ===
Spark session dihentikan
```

4.1 Script ML di PySpark (lanjutan dari step 3.3)

```
# Import ML libraries
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml import Pipeline

# Baca data dari Silver Layer
df_ml = spark.read.option("header", "true").option("inferSchema",
"true").csv("hdfs://namenode:9000/bigdata/silver/pertanian/data_clean")

# Persiapan fitur untuk ML
feature_cols = ["luas_panen_ha", "curah_hujan_mm", "kelembapan_%", "suhu_celsius"]
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")

# Buat model Random Forest
rf = RandomForestRegressor(featuresCol="features", labelCol="produksi_ton",
```

numTrees=10)

# Pipeline
pipeline = Pipeline(stages=[assembler, rf])

Split data training dan testing
(training_data, test_data) = df_ml.randomSplit([0.8, 0.2], seed=42)

# Training model
model = pipeline.fit(training_data)

# Prediksi
predictions = model.transform(test_data)

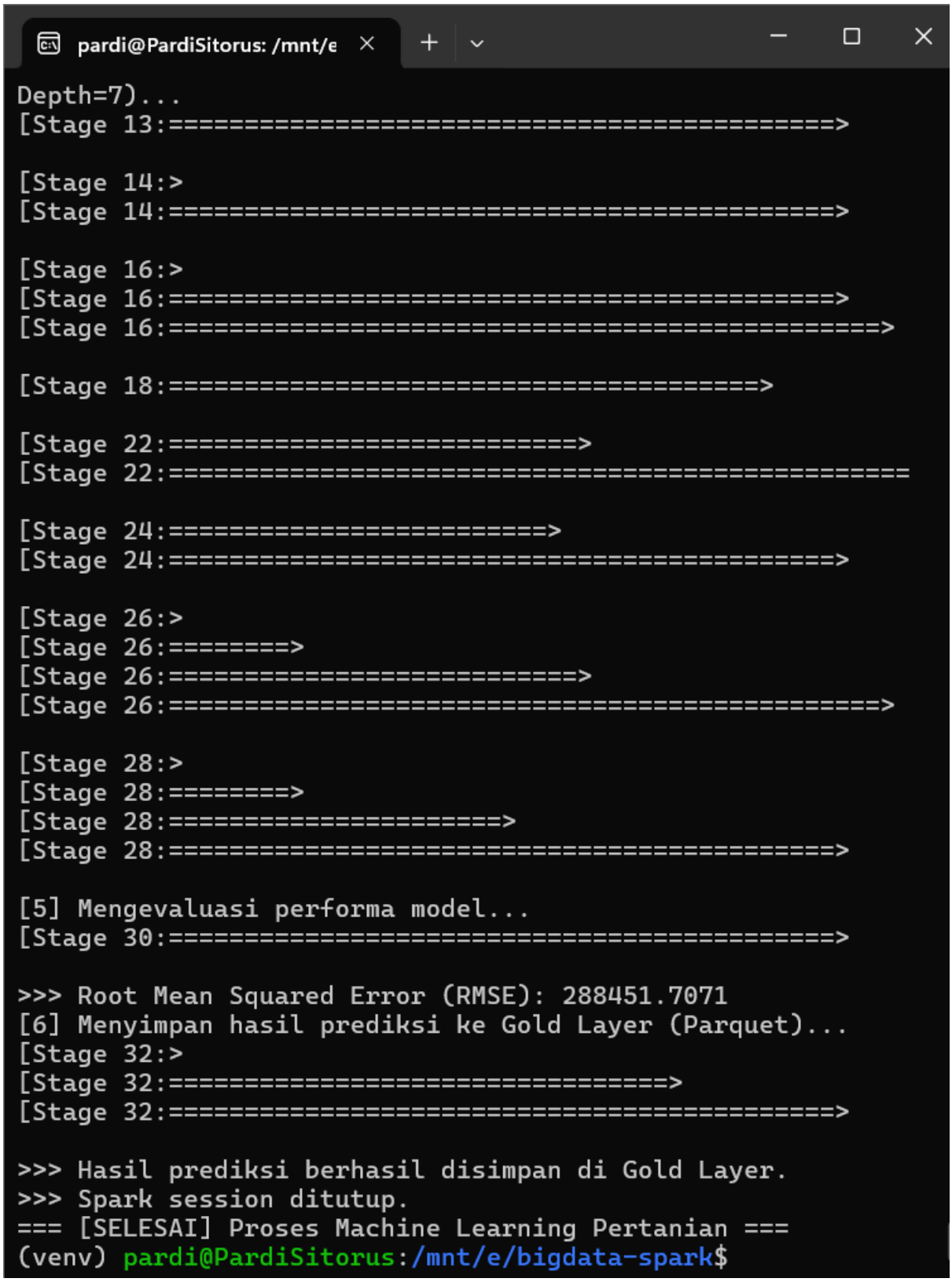
# Evaluasi model
evaluator = RegressionEvaluator(labelCol="produksi_ton", predictionCol="prediction",
metricName="rmse")
rmse = evaluator.evaluate(predictions)
print(f"Root Mean Squared Error: {rmse}")

Simpan hasil prediksi ke Gold Layer
predictions.select("provinsi", "tahun", "komoditas", "produksi_ton",
"prediction").coalesce(1).write.mode("overwrite").option("header",
"true").csv("hdfs://namenode:9000/bigdata/gold/pertanian/predictions")

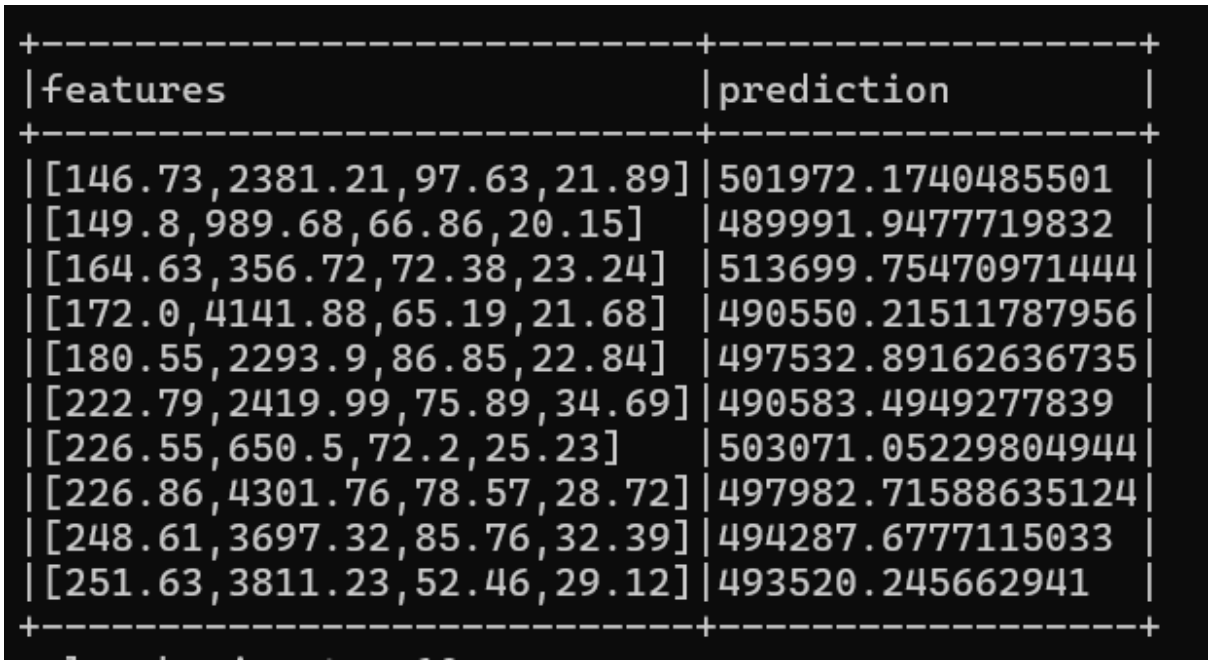
Install python

```
root@localhost:/# python3
Python 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
t sklearn

print(numpy.__version__)
print(pandas.__version__)
print(sklearn.__version__)
>>> import pandas
>>> import sklearn
>>>
>>> print(numpy.__version__)
2.2.6
>>> print(pandas.__version__)
2.2.3
>>> print(sklearn.__version__)
```

```
25/05/25 09:13:39 INFO DAGScheduler: ResultStage 4 (count at NativeMethodAcc
essorImpl.java:0) finished in 0.141 s
25/05/25 09:13:39 INFO DAGScheduler: Job 3 is finished. Cancelling potential
 speculative or zombie tasks for this job
25/05/25 09:13:39 INFO TaskSchedulerImpl: Killing all running tasks in stage
 4: Stage finished
25/05/25 09:13:39 INFO DAGScheduler: Job 3 finished: count at NativeMethodAc
cessorImpl.java:0, took 0.155963 s
Total record awal: 1048575
Schema data:
root
 |-- provinsi: string (nullable = true)
 |-- tahun: integer (nullable = true)
 |-- komoditas: string (nullable = true)
 |-- produksi_ton: double (nullable = true)
 |-- luas_panen_ha: double (nullable = true)
 |-- curah_hujan_mm: double (nullable = true)
 |-- kelembapan_%: double (nullable = true)
 |-- suhu_celsius: double (nullable = true)

Contoh data (5 baris pertama):
25/05/25 09:13:39 INFO FileSourceStrategy: Pushed Filters:
25/05/25 09:13:39 INFO FileSourceStrategy: Post-Scan Filters:
25/05/25 09:13:39 INFO CodeGenerator: Code generated in 17.116545 ms
25/05/25 09:13:39 INFO MemoryStore: Block broadcast_7 stored as values in me
mory (estimated size 199.6 KiB, free 433.7 MiB)
25/05/25 09:13:39 INFO MemoryStore: Block broadcast_7_piece0 stored as bytes
 in memory (estimated size 34.3 KiB, free 433.6 MiB)
25/05/25 09:13:39 INFO BlockManagerInfo: Added broadcast_7_piece0 in memory
on 172.19.234.49:41121 (size: 34.3 KiB, free: 434.3 MiB)
25/05/25 09:13:39 INFO SparkContext: Created broadcast 7 from showString at
```



Membaca hasil prediksi



```
+--------------------------+-----------------+
|features                  |prediction       |
+--------------------------+-----------------+
|[146.73,2381.21,97.63,21.89]|501972.1740485501 |
|[149.8,989.68,66.86,20.15]  |489991.9477719832 |
|[164.63,356.72,72.38,23.24] |513699.75470971444|
|[172.0,4141.88,65.19,21.68] |490550.21511787956|
|[180.55,2293.9,86.85,22.84] |497532.89162636735|
|[222.79,2419.99,75.89,34.69]|490583.4949277839 |
|[226.55,650.5,72.2,25.23]   |503071.05229804944|
|[226.86,4301.76,78.57,28.72]|497982.71588635124|
|[248.61,3697.32,85.76,32.39]|494287.6777115033 |
|[251.63,3811.23,52.46,29.12]|493520.245662941  |
+--------------------------+-----------------+
```

print("Model berhasil dilatih dan prediksi disimpan ke Gold Layer")
STEP 5: Export Data untuk Power BI
5.1 Export ke format yang mudah dibaca Power BI

```
# Baca data prediksi
df_export = spark.read.option("header", "true").option("inferSchema",
"true").csv("hdfs://namenode:9000/bigdata/gold/pertanian/predictions")

# Tambah kolom analisis sederhana
df_export_final = df_export.withColumn("akurasi_prediksi",
    when(abs(col("produksi_ton") - col("prediction")) / col("produksi_ton") < 0.1, "Sangat
Akurat")
    .when(abs(col("produksi_ton") - col("prediction")) / col("produksi_ton") < 0.2, "Akurat")
    .otherwise("Kurang Akurat"))

# Tampilkan hasil
df_export_final.show(10)

# Keluar dari PySpark
exit()
```

STEP 6: Setup Hive untuk Power BI
6.1 Masuk ke Hive container

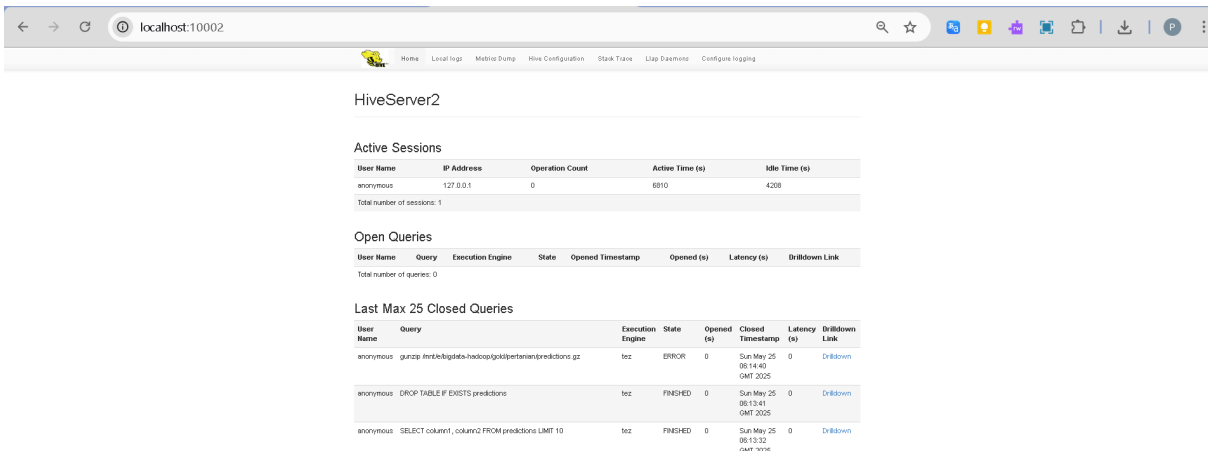
```
# Terminal baru
docker exec -it bigdata-hive bash
```

6.2 Buat tabel Hive

Jalankan Hive CLI
Hive

```
pardi@PardiSitorus:/mnt/e/bigdata-hive-mysql$ ^C
pardi@PardiSitorus:/mnt/e/bigdata-hive-mysql$ # Terminal baru
docker exec -it bigdata-hive bash
root@localhost:/# ^C
root@localhost:/# # Jalankan Hive CLI
hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/apache-hive-4.0.1-bin/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apache-tez-0.10.4-bin/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/hadoop-3.4.1/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
WARNING: Use "yarn jar" to launch YARN applications.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/apache-hive-4.0.1-bin/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/apache-tez-0.10.4-bin/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/hadoop-3.4.1/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Beeline version 4.0.1 by Apache Hive
beeline>
```

6.3 Script Hive (di dalam hive CLI)



-- Buat database
CREATE DATABASE IF NOT EXISTS pertanian;
USE pertanian;

anonymous	USE pertanian	tez	FINISHED	0	Sun May 25 05:30:57 GMT 2025	0	Drilldown
anonymous	CREATE DATABASE IF NOT EXISTS pertanian	tez	FINISHED	0	Sun May 25 05:30:57 GMT 2025	0	Drilldown

Total number of queries: 25

```
-- Buat tabel eksternal untuk data prediksi
CREATE EXTERNAL TABLE predictions_table (
    provinsi STRING,
    tahun INT,
    komoditas STRING,
    produksi_ton DOUBLE,
    prediction DOUBLE
)
```

anonymous	CREATE EXTERNAL TABLE predictions_table (provinsi STRING, tahun INT, komoditas STRING, produksi_ton DOUBLE, prediction DOUBLE)	tez	FINISHED	0	Sun May 25 06:07:05 GMT 2025	0	Drilldown

```
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 'hdfs://namenode:9000/bigdata/gold/pertanian/predictions'
TBLPROPERTIES ("skip.header.line.count"="1");

-- Test query
SELECT * FROM predictions_table LIMIT 10;
```

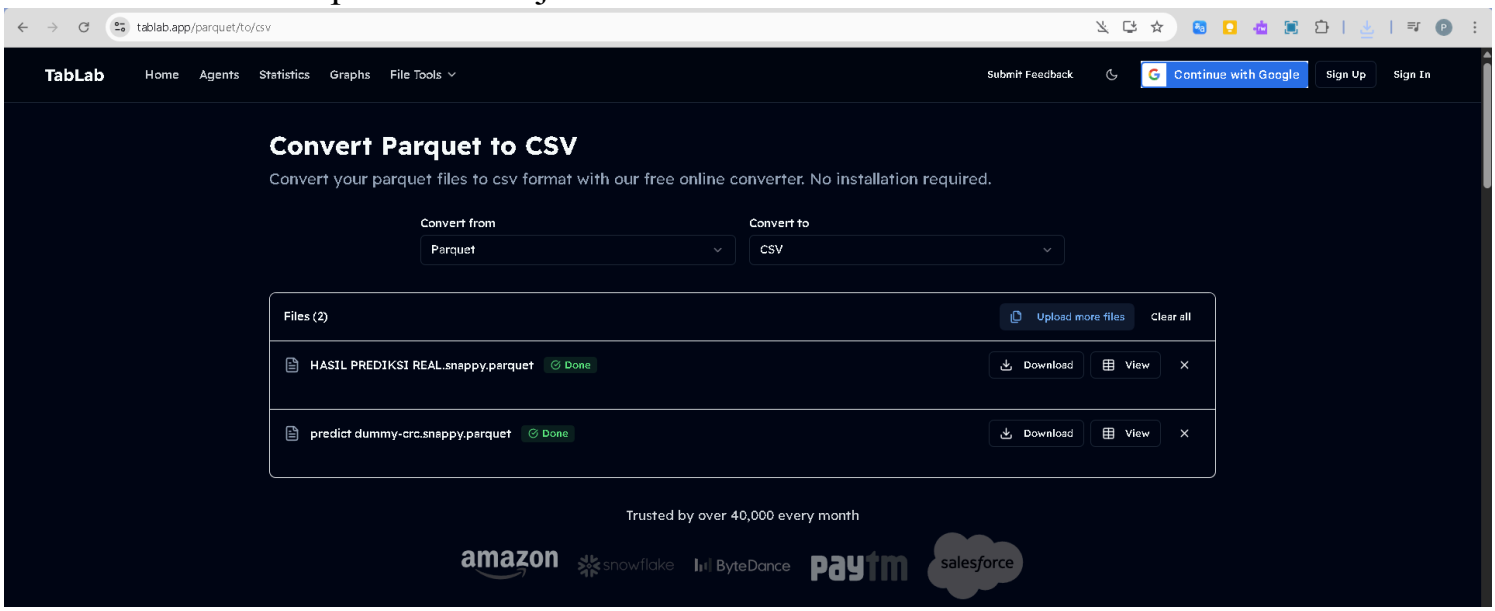
anonymous	SELECT column1, column2 FROM predictions LIMIT 10	tez	FINISHED	0	Sun May 25 06:13:32 GMT 2025	0	Drilldown

```
-- Buat summary table untuk dashboard
CREATE TABLE summary_predictions AS
SELECT
    provinsi,
    komoditas,
    AVG(produksi_ton) as avg_produksi_aktual,
    AVG(prediction) as avg_produksi_prediksi,
    COUNT(*) as jumlah_record
FROM predictions_table
GROUP BY provinsi, komoditas;


SELECT * FROM summary_predictions;
```

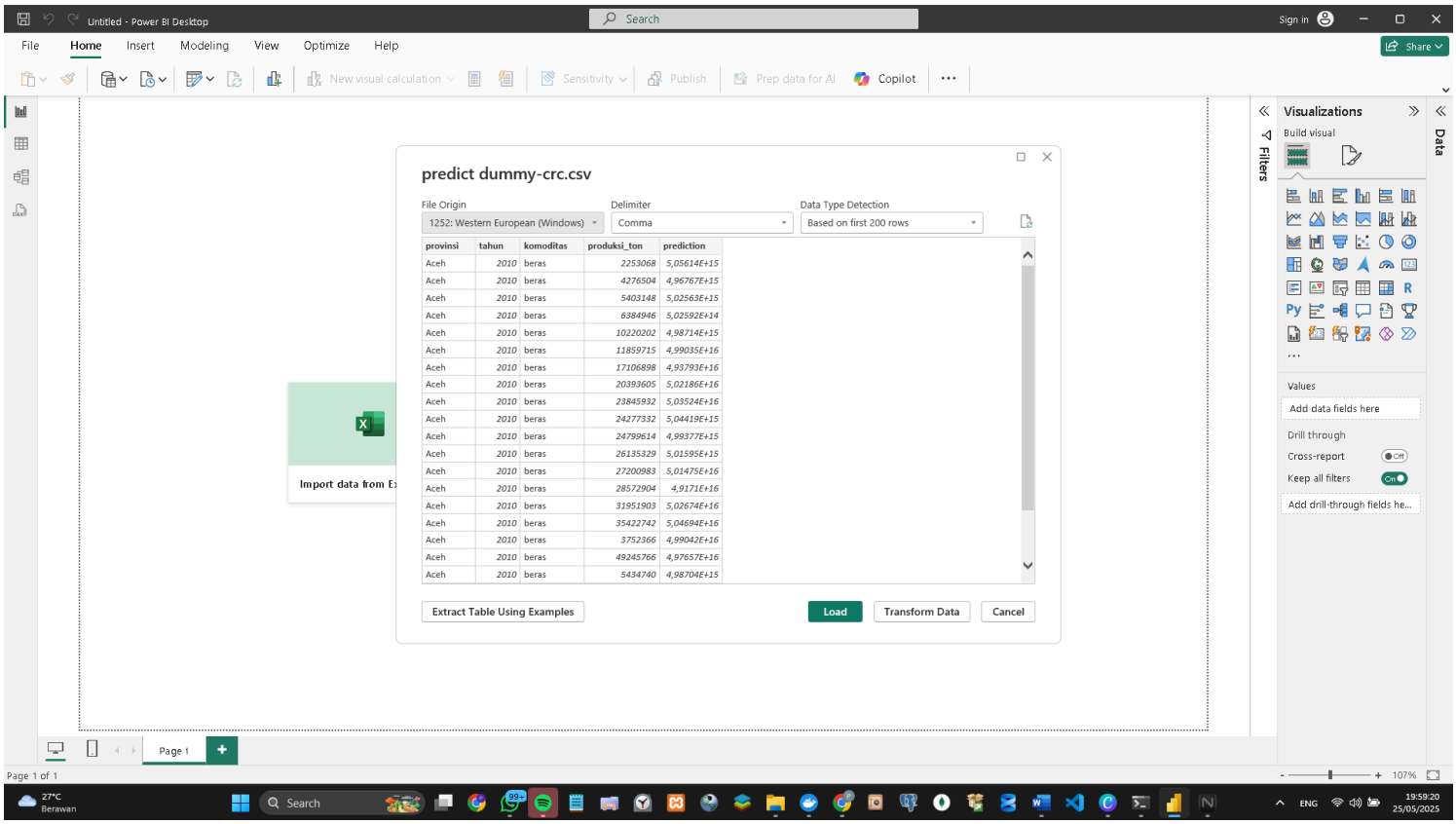
User Name	Query	Execution Engine	State	Opened (s)	Closed Timestamp	Latency (s)	Drilldown Link
hive	SELECT * FROM summary_predictions	tez	FINISHED	0	Sun May 25 12:52:59 GMT 2025	0	Drilldown

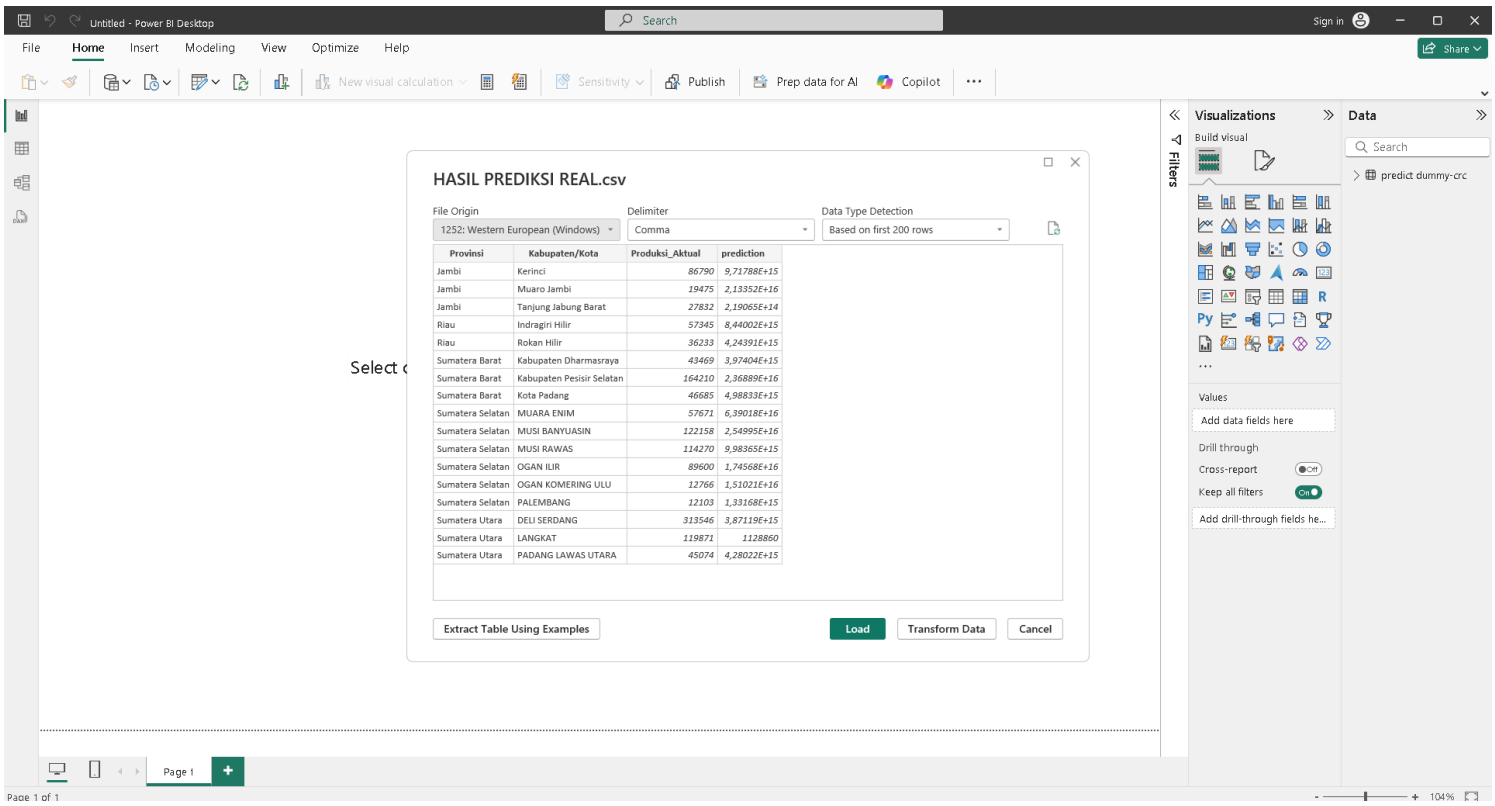
STEP 7: Verifikasi dan Download Data
7.1 Convert Parquet Menjadi Data CSV
Keluar dari hive (ketik exit)
exit

Convert data hasil prediksi menjadi csv



STEP 8: Koneksi Power BI
8.1 Insert Data Prediksi CSV ke Power BI





8.2 Query untuk Dashboard

```
-- Query untuk visualisasi produksi per provinsi
SELECT provinsi, SUM(produksi_ton) as total_produksi
FROM predictions_table
GROUP BY provinsi;
```

-- Query untuk perbandingan aktual vs prediksi
SELECT provinsi, komoditas, produksi_ton, prediction
FROM predictions_table;
Troubleshooting Cepat